What is claimed is:

1. A computer system for executing predicated instructions wherein each instruction includes a guard, the value of which determines whether or not that instruction is executed, the computer system comprising:

a fetch unit for fetching instructions to be executed;

a decode unit for decoding said instructions;

at least one pipelined execution unit for executing decoded instructions and being associated with a guard register file holding values of the guards to allow resolution of the guards to be made; and

an emulation unit including control circuitry which cooperates with the decode unit to selectively control the decode unit to implement a precise watch or a non-precise watch on detection of a breakpoint wherein according to a precise watch, the instruction causing the breakpoint is held at the decode unit and, according to a non-precise watch, the instruction causing the breakpoint and subsequent instructions are permitted to be supplied and from the decode unit to the at least one execution unit while guard resolution in said at least one execution pipeline is awaited.

2. A computer system according to claim 1, which is implemented on a single chip.

3. A computer system according to claim 1, which includes a program memory for holding said instructions to be executed.

4. A computer system according to claim 1, wherein the emulation unit is associated with an emulation program memory which holds debug code which is executed in a debug mode.

5. A computer system according to claim 1, wherein, when the emulation unit is in a precise watch mode, it is operable to issue a request to the execution pipeline for guard resolution, the guard resolution being transmitted to the control circuitry

of the emulation unit which is responsive thereto to control operation of the decode unit.

6. A computer system according to claim 5, wherein the control circuitry of the emulation unit is operable to issue one of a go command and divert command to the decode unit responsive to receipt of the guard resolution from the execution pipeline, wherein a go command allows the instruction which caused the breakpoint and subsequent instructions to be normally decoded and executed, and a divert command sets the computer system into a debug mode.

7. A computer system according to claim 1, wherein, when the emulation unit is in non-precise watch mode, the instruction which caused the breakpoint and subsequent instructions are decoded and executed normally until such time as said instruction reaches the execution pipeline where its guard is resolved such that a commit signal is generated to the control circuitry of the emulation unit, and wherein the emulation unit is responsive to receipt of the commit signal to set the computer system into a debug mode.

8. A computer system according to claim 1, which includes a microinstruction generator which receives instructions from the decode unit and supplies microinstructions to the execution pipeline, said microinstructions including fields for holding respective guards to be resolved.

9. A computer system according to claim 1, which includes a plurality of parallel pipelined execution units, including at least two data unit pipelines for executing data processing instructions and at least two address unit pipelines for executing memory access instructions.

10. A method of debugging an on-chip processor which is arranged to execute predicated instructions wherein each instruction includes a guard, the value of which determines whether or not

that instruction is executed, the method comprising:

fetching instructions to be executed;

decoding said instructions;

executing decoded instructions, said executing step including resolving values of the guards of the instructions; and

detecting instructions which have a debug effect and acting on said instructions in dependence on whether the processor is in a precise watch mode or a non-precise watch mode wherein, according to a precise watch mode, the instruction is not decoded and, according to a non-precise watch mode, the instruction and subsequent instruction are supplied and executed normally while guard resolution is awaited.

11. A method according to claim 10, wherein breakpoints *are* detected at instructions having certain program count*s*.

12. A method according to claim 10, wherein breakpoint*s* are detected at instructions having certain opcode*s*.

13. A method according to claim 10, wherein, in a precise watch mode, a request for guard resolution is issued such that an instruction guard is resolved prior to execution of the instruction, selectively causing issue of one of a go command and a debug command responsive to the guard resolution.

14. A method according to claim 12, wherein, in a debug mode, debug code is executed by the processor.

15. A method according to claim 10, wherein, in a non-precise watch mode, the instruction which caused the breakpoint and subsequent instructions are decoded and executed normally until such time as the guard of the breakpoint instruction has been resolved wherein, if the guard is resolved such that a position commit signal is generated, the processor is set into a debug mode.